**CSC 1100 – Problem Solving and Programming**

**Project 2 – Rob Kokochak**

**50 points – Due April 29, 11am**

**Late deadline is May 1, 11:59pm, but 20% off**

**a)** Save this document with your name and the project number somewhere in the file name.

**b)** Type/paste your answers into the document.

**c)** Submit the following documents to the Canvas assignment link where you downloaded this document:

✓ This document.

✓ Your .cpp files renamed to .txt.

Submit the documents separately, not as one .zip file.

You've been hired again by Part Pretenders to enhance the great C++ console application you wrote for Labs 20 and 22. Expand the string array **parts** size from ten to 100 but remove the initializer list. The parts will now come from an input file. Create text file **CarParts.txt** and place at least fifteen car parts in array **parts**. The file contains no header line. Each detail line contains one car part. Each part name contains no spaces (you can use underscore to separate words within a part name). The part names should not be sorted within the file. Place the file in a folder where your development tool can locate it, or place the file in a data folder and use a path to the file. Create or modify the following functions:

**void addPart(string parts[], int &partCount)**

After a part is added to array **parts**, sort its data. This ensures that the data is always in alphabetical order.

**void deletePart(string parts[], int &partCount)**

Add logic to prompt the user for the index of the car part to delete. If the index is invalid (<0 or >= partCount), print an error message. If the index is valid, remove the part from array **parts**. See the Homework 6 key for code to do this. Change the message to indicate "part at index removed".

**void listParts(string parts[], int partCount)**

Use formatted output manipulators (setw, left/right) to print in one row the following:

● Index

● Part name

**int menuOption()**

Change the menu to:

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: ";

Ensure the correct function is called:

**1 – List car parts** calls function listParts.

**2 – Search car parts** calls function searchParts.

**3 – Add part** calls function addPart.

**4 – Delete part** calls function deletePart.

**int readParts(string parts[])**

This value function attempts to open the input file. If it doesn't open, return a negative number. If it does open, print a file-reading message, loop through the file and store each part read in array **parts**, print the number of lines read, and return the line count.

**int main()**

Call function readParts and test the returned value. If the file didn't open, print an error message. If the file opened, sort array parts, and continue to read and process menu options until the user enters the sentinel value of 9.

Define constants for the input file name, array size, and the column widths. Insure that your code adheres to the styles as defined in document **Source code styles** on Canvas. Provide a complete header comment and body comments. Arrange the programmer-defined functions in alphabetical order and place function **main** at the end. The output should look like:

Welcome to Part Pretenders

--------------------------

Reading lines from file 'CarParts.txt' ...

15 line(s) read from file 'CarParts.txt'.

Parts sorted.

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 1

Part List

Index Part

0 Air\_bag

…

14 Transmission

Part count: 15

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 2

Enter a part to search for (no spaces): Gearshift

'Gearshift' NOT found.

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 3

Enter the part to add: Gearshift

Part 'Gearshift' added to list.

Parts sorted.

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 1

Part List

Index Part

0 Air\_bag

..

9 Gearshift

…

12 Rear\_bumper

13 Side\_mirror

14 Tire

15 Transmission

Part count: 16

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 4

Enter index of part to delete: 22

Error: invalid index of 22.

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 4

Enter index of part to delete: 13

Part at index 13 deleted from list.

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 1

Part List

Index Part

0 Air\_bag

…

12 Rear\_bumper

13 Tire

14 Transmission

Part count: 15

Part Pretenders Menu

1 - List car parts

2 - Search car parts

3 - Add car part

4 - Delete car part

9 - Exit

Enter an option: 9

End of Part Pretenders

Run the program using the following menu option order:

1

2

3

1

3

1

4

1

4

1

2

*[your program code here]\**

**//==========================================================**

**//**

**// Title: Part Pretenders, v3**

**// Course: CSC 1100**

**// Lab Number: Project 2**

**// Author: Rob Kokochak**

**// Date: April 25 2021**

**// Description:**

**//**

**// This C++ console application reads car parts from an**

**// external .txt file, sorts them alphabetically and puts**

**// them into an array. It then allows for various processing**

**// options on the data via a menu option such as listing the parts,**

**// searching for a part, adding a new part or deletig a part.**

**//==========================================================**

**#include <cstdlib> // For several general-purpose functions**

**#include <fstream> // For file handling**

**#include <iomanip> // For formatted output**

**#include <iostream> // For cin, cout, and system**

**#include <string> // For string data type**

**using namespace std; // So "std::cout" may be abbreviated to "cout"**

**//==========================================================**

**// Constants**

**//==========================================================**

**const string INPUT\_FILE\_NAME = "CarParts.txt";**

**const int ARRAY\_SIZE = 100, COLW1 = 10, COLW2 = 20;**

**// Function Prototypes**

**void addPart(string parts[], int &partCount);**

**int binarySearchParts(string parts[], int arraySize, string key);**

**void deletePart(string parts[], int &partCount);**

**void listParts(string parts[], int arraySize);**

**int menuOption();**

**void searchParts(string parts[], int arraySize);**

**void sortParts(string parts[], int arraySize);**

**//==========================================================**

**// Function Definitions**

**//==========================================================**

**//==========================================================**

**// addPart**

**//==========================================================**

**void addPart(string parts[], int &partCount)**

**{**

**// Declare variables**

**string part;**

**// Test whether room to add part**

**if (partCount == ARRAY\_SIZE)**

**cout << "No room to add part at this time." << endl;**

**else**

**{**

**// Prompt for and get part**

**cout << "Enter the part to add: ";**

**cin >> part;**

**// Add part**

**parts[partCount] = part;**

**partCount = partCount + 1;**

**cout << "Part '" << part << "' added to list.";**

**sortParts(parts, partCount);**

**}**

**}**

**//==========================================================**

**// binarySearchParts**

**//==========================================================**

**int binarySearchParts(string parts[], int arraySize, string key)**

**{**

**// Declare variables**

**int min = 0;**

**int index;**

**int max = arraySize - 1;**

**// Loop to find key**

**while (min <= max)**

**{**

**// Get midpoint of array**

**index = (min + max) / 2;**

**// Test if key found**

**if (parts[index] == key) // Guess is right on**

**return index;**

**else if (parts[index] < key) // Guess is too low**

**min = index + 1;**

**else**

**max = index - 1;**

**}**

**// Return index**

**return -1;**

**}**

**//==========================================================**

**// deletePart**

**//==========================================================**

**void deletePart(string parts[], int &partCount)**

**{**

**// Test whether room to add part**

**if (partCount == 0)**

**cout << "No part to delete at this time." << endl;**

**else**

**{**

**int index;**

**cout << "Enter a part's index to delete it: ";**

**cin >> index;**

**// Test validity**

**if (index < 0 || index >= partCount)**

**cout << "Error: invalid index of " << index << "." << endl;**

**else{**

**cout << "Part at index " << index << ": '" << parts[index]**

**<< "' removed from list." << endl;**

**// Delete part**

**for (int i = index; i < partCount - 1; i++)**

**parts[i] = parts[i + 1];**

**partCount = partCount - 1;**

**}**

**}**

**}**

**//==========================================================**

**// listParts**

**//==========================================================**

**void listParts(string parts[], int arraySize)**

**{**

**cout << endl << "Part List" << endl;**

**cout << "---------" << endl;**

**// Loop to list parts**

**for (int i = 0; i < arraySize; i++)**

**cout << setw(COLW1) << left << ("Index " + to\_string(i) + ": ") <<**

**setw(COLW2) << left << parts[i] << endl;**

**cout << endl << "Part count: " << arraySize << endl;**

**}**

**//==========================================================**

**// menuOption**

**//==========================================================**

**int menuOption()**

**{**

**// Declare variables**

**int option;**

**// Show menu and get option**

**cout << "\nPart Pretenders Menu" << endl;**

**cout << "1 - List car parts" << endl;**

**cout << "2 - Search car parts" << endl;**

**cout << "3 - Add car part" << endl;**

**cout << "4 - Delete car part" << endl;**

**cout << "9 - Exit" << endl;**

**cout << "\nEnter an option: ";**

**cin >> option;**

**return option;**

**}**

**//==========================================================**

**// readParts**

**//==========================================================**

**int readParts(string parts[])**

**{**

**//declare variables**

**ifstream inFile;**

**int lineCount = 0;**

**string line;**

**// Attempt to open file**

**inFile.open(INPUT\_FILE\_NAME);**

**if (!inFile.is\_open())**

**{**

**cout << "Error: unable to open file '"**

**<< INPUT\_FILE\_NAME << "'." << endl << endl;**

**}**

**else**

**{**

**cout << "Reading from file " << "'" << INPUT\_FILE\_NAME << "' ..." << endl;**

**// Populate array parts and update line count**

**lineCount = 0;**

**for (int i = 0; inFile.good(); i++)**

**{**

**getline(inFile, line);**

**parts[i] = line;**

**lineCount = lineCount + 1;**

**}**

**cout << lineCount << " line(s) read from file " << "'" <<**

**INPUT\_FILE\_NAME << "'.";**

**// return lineCount**

**return lineCount;**

**}**

**// return -1 if file wasn't able to open**

**return -1;**

**}**

**//==========================================================**

**// searchParts**

**//==========================================================**

**void searchParts(string parts[], int arraySize)**

**{**

**// Declare variables**

**string key;**

**int index;**

**// Prompt**

**cout << "\nEnter a part to search for (no spaces): ";**

**cin >> key;**

**// Perform search**

**index = binarySearchParts(parts, arraySize, key);**

**// Display Result**

**if (index != -1)**

**cout << "'" << key << "' found at index "**

**<< index << "." << endl;**

**else**

**cout << "'" << key << "' NOT found." << endl;**

**}**

**//==========================================================**

**// sortParts**

**//==========================================================**

**void sortParts(string parts[], int arraySize)**

**{**

**// Declare variables**

**string value;**

**int spot;**

**// Loop to test each value**

**for (int i = 1; i < arraySize; i++)**

**{**

**// Loop to find spot to place value**

**value = parts[i];**

**spot = i - 1;**

**while (spot >= 0 && parts[spot] > value)**

**{**

**parts[spot + 1] = parts[spot];**

**spot = spot - 1;**

**}**

**// Place value in spot**

**parts[spot + 1] = value;**

**}**

**// Print sorted message**

**cout << endl << "Parts sorted." << endl;**

**}**

**//==========================================================**

**// main**

**//==========================================================**

**int main()**

**{**

**// Show application header**

**cout << "Welcome to Part Pretenders" << endl;**

**cout << "--------------------------" << endl << endl;**

**string parts[ARRAY\_SIZE];**

**// Populate array parts with items from CarParts.txt**

**int partCount = readParts(parts);**

**sortParts(parts, partCount);**

**// Loop to process options**

**int option = menuOption();**

**while (option != 9)**

**{**

**// Handle option**

**switch (option)**

**{**

**// List parts**

**case 1:**

**listParts(parts, partCount);**

**break;**

**// Search parts**

**case 2:**

**searchParts(parts, partCount);**

**break;**

**// Add part**

**case 3:**

**addPart(parts, partCount);**

**break;**

**// Delete part**

**case 4:**

**deletePart(parts, partCount);**

**break;**

**// Handle invalid option**

**default:**

**cout << endl << "Error: unknown option of '" << option**

**<< "'." << endl;**

**}**

**// Get next option**

**option = menuOption();**

**}**

**// Show application close**

**cout << "\nEnd of Part Pretenders" << endl;**

**}**

*[your CarParts.txt here]\*\**

Differential

Transmission

Shift\_Lever

Steering\_Wheel

Wheels

Tires

Bumper

Headlights

Engine

Seats

Radio

Seatbelts

Hood

Body

Mirrors

*[your program output here]\*\**

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

**\* Copying-and-pasting C++ code to a Word document**

**macOS**

1) From within the C++ program, press **command-A** and press **command-C**.

2) From within the Word document, press **command-V**.

**Windows**

1) From within the C++ program, press **CTRL-A** and press **CTRL-C**.

2) From within the Word document, press **CTRL-V**.

**\*\* Copying-and-pasting C++ console application output to a Word document**

**macOS**

1) From the C++ console, press **shift-command-4-space**.

2) From within the Word document, **command-V**.

**Windows**

1) From the C++ console, press **ALT-PrintScreen**.

2) From within the Word document, press **CTRL-V**.